Graphics Data

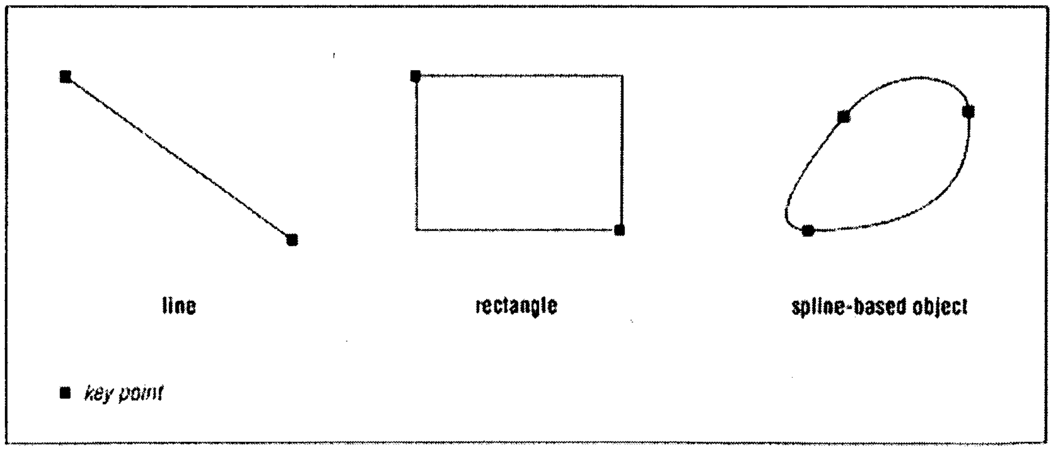
Graphics data is traditionally divided into two classes: *vector* and *bitmap*. As we explain below, we use the term *bitmap* to replace the older term *raster*.

Vector Data

In computer graphics, *vector data* usually refers to a means of representing lines, polygons, or curves (or any object that can be easily drawn with lines) by numerically specifying *key points*. The job of a program rendering this key-point data is to regenerate the lines by somehow connecting the key points or by drawing using the key points for guidance. Always associated with vector data is attribute information (such as color and line thickness information) and a set of conventions (or rules) allowing a program to draw the desired objects. These conventions can be either implicit or explicit, and, although designed to accomplish the same goals, are generally different from program to program.

Figure 1-2 shows several examples of vector data.

Figure 1-2: Vector data



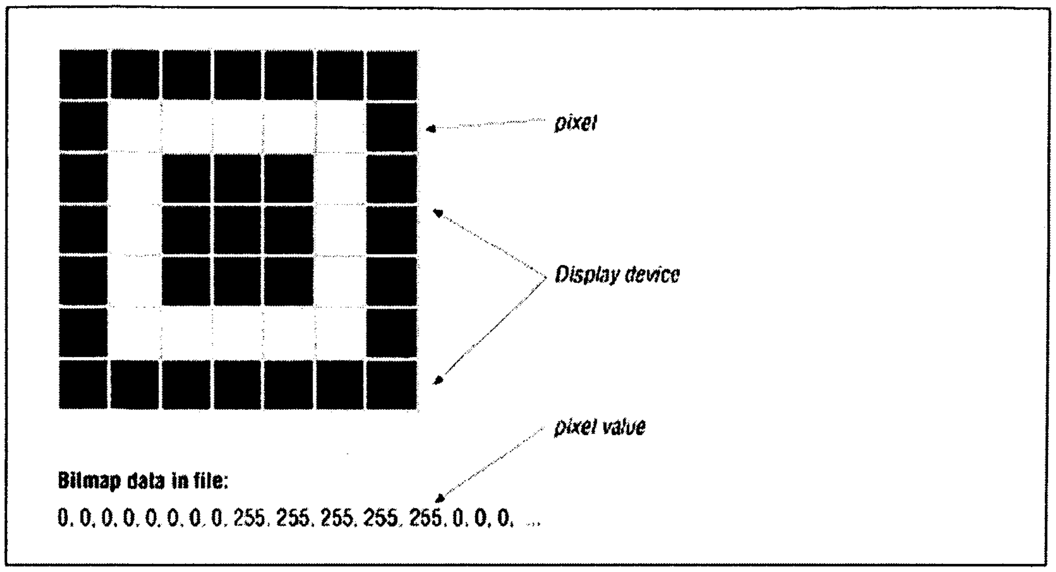
By the way, you may be familiar with a definition of the word *vector* which is quite precise. In the sciences and mathematics, for instance, a vector is a straight line having both magnitude and direction. In computer graphics, *vector* is a sort of catch-all term. It can be almost any kind of line or line segment, and it is usually specified by sets of endpoints, except in the case of curved lines and more complicated geometric figures, which require other key points to be fully specified.

Bitmap Data

Bitmap data is formed from a set of numerical values specifying the colors of individual *pixels* or *picture elements* (*pels*). Pixels are dots of color arranged on a regular grid in a pattern representing the form to be displayed. We commonly say that a bitmap is an *array of pixels*, although a bitmap, technically, consists of an *array of numerical values* used to set, color, or "turn on" the corresponding pixels on an output device when the bitmap is rendered. If there is any ambiguity in the text, we will make the distinction clear by using the term *pixel value* to refer to a numerical value in the bitmap data corresponding to a pixel color in the image on the display device.

Figure 1-3 shows an example of bitmap data.

Figure 1-3: Bitmap data



In older usage, the term *bitmap* sometimes referred to an array (or "map") of single bits, each bit corresponding to a pixel, while the terms *pixelmap*, *graymap*, and *pixmap* were reserved for arrays of multibit pixels. We always use the term *bitmap* to refer to an array of pixels, whatever the type, and specify the *bit depth*, or *pixel depth*, which is the size of the pixels in bits or some other convenient unit (such as bytes). The bit depth determines the number of colors a pixel value can represent. A 1-bit pixel can be one of two colors, a 4-bit pixel one of 16 colors, and so on. The most commonly found pixel depths today are 1, 2, 4, 8, 15, 16, 24, and 32 bits. Some of the reasons for this, and other color-related topics, are discussed in Chapter 2, *Computer Graphics Basics*.

Sources of Bitmap Data: Raster Devices

Historically, the term *raster* has been associated with cathode ray tube (CRT) technology and has referred to the pattern of rows the device makes when displaying an image on a picture tube. Raster-format images are therefore a collection of pixels organized into a series of rows, which are called *scan lines*. Because raster output devices, by far the most popular kind available today, display images as patterns of pixels, pixel values in a bitmap are usually arranged so as to make them easy to display on certain common raster devices. For these reasons, bitmap data is often called *raster data*. We use the term *bitmap data* .

As mentioned above, bitmap data can be produced when a program renders graphics data and writes the corresponding output image to a file instead of displaying it on an output device. This is one of the reasons bitmaps and bitmap data are often referred to as *images*, and bitmap data is referred to as *image data*. Although there is nothing to see in the traditional sense, an image can be readily reconstructed from the file and displayed on an output device. We will occasionally refer to the block of pixel values in a bitmap file as the *image* or *image portion*.

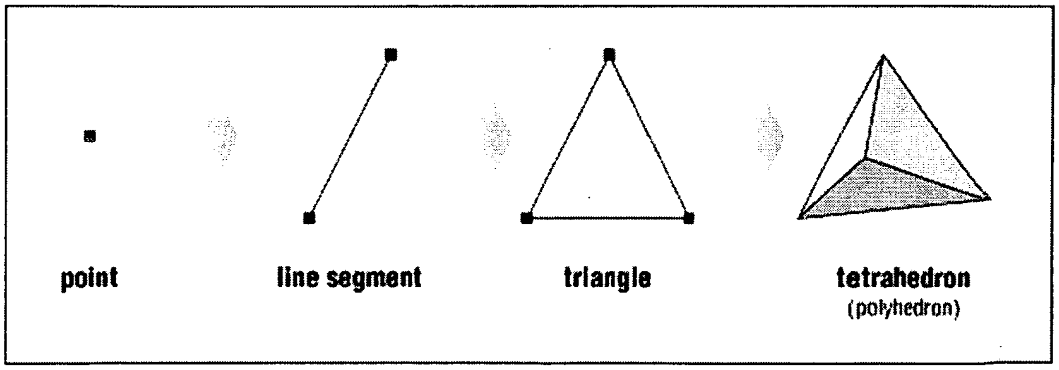
Other sources of bitmap data are raster devices used to work with images in the traditional sense of the word; raster devices are scanners, video cameras, and other digitizing devices. For our purposes, we consider a raster device that produces digital data to be just another source of graphics data, and we say that the graphics data is rendered when the program used to capture the data from the device writes it to a file. When speaking of graphics data captured from a real-world source, such as a scanner, people speak redundantly of a *bitmap image*, or an *image bitmap.*

What About Object Data?

People sometimes speak of a third class: *object data.* In the past, this referred to a method of designating complex forms, such as nested polygons, through a shorthand method of notation, and relying on a program's ability to directly render these forms with a minimal set of clues. Increasingly, however, the term is used to refer to data stored along with the program code or algorithmic information needed to render it. This distinction may become useful in the future, particularly if languages that support object-oriented programming (such as Smalltalk and C++) become more popular. However, for now we choose to ignore this third primitive data type, mainly because at the time of this writing, there are no standardized object file formats of any apparent commercial importance. In any case, the data portions of all objects can be decomposed into simpler forms composed of elements from one of the two primitive classes.

Figure 1-4 shows an example of object data.

Figure 1-4: Object data



Other Data

Graphics files may also include data containing structural, color, and other descriptive information. This information is included primarily as an aid to the rendering application in reconstructing and displaying an image.

From Vector to Bitmap Data...

Twenty-five years ago, computer graphics was based almost entirely on vector data. Random-scan vector displays and pen plotters were the only readily obtainable output devices. The advent of cheap, high-capacity magnetic media, in the form of tapes and disks, soon allowed the storage of large files, which in turn created a need for the first standardized graphics file formats.

Today, most graphics storage is bitmap-based, and displays are raster-based. This is due in part to the availability of high-speed CPUs, inexpensive memory and mass storage, and high-resolution input and output hardware. Bitmap graphics are also driven by the need to manipulate images obtained from raster digitizing devices. Bitmap graphics are important in applications supporting CAD and 3D rendering, business charts, 2D and 3D modeling, computer art and animation, graphical user interfaces, video games, electronic document image processing (EDIP), and image processing and analysis.

It is interesting to note that the increased emphasis on bitmap graphics in our time corresponds to a shift toward the output end of the graphics production pipeline. By volume, the greatest amount of data being stored and exchanged consists of finished images in bitmap format.

The explosive growth of the World Wide Web has fueled this shift. Almost every Web page has one or more bitmap files associated with it. Bitmap images have become a part of everyday life for millions of people.

...And Back Again

But the trend toward bitmap data may not last. Although there are certain advantages to storing graphics images as bitmap data (we cover these in the section called "Pros and Cons of Bitmap File Formats" in Chapter 3, *Bitmap Files*), bitmap images are usually pretty bulky. There is a definite trend toward networking in all the computer markets, and big bitmap files and low-cost networks don't mix. The cost of sending files around on the Internet, for example, can be measured not only in connect costs, but in lost time and decreased network performance.

The growth of the World Wide Web has accelerated this trend. The Web is currently built around HTML, a text markup language allowing software run on the machines of remote users to construct elaborate images of text pages with only minimal clues. This strategy, one of offloading imaging tasks to the machines of remote users to conserve network bandwidth, is being pursued by a number of vendors. This work is typified by the development of Java, Sun Microsystem's Internet programming language.

Because graphics files are surely not going to go away, we expect some sort of vector-based file format to emerge somewhere along the line as an interchange standard. Unfortunately, none of the present vector formats in common use is acceptable to a broad range of users (but this may change).

PREVIOUS NEXT UP TO INDEX
CONTENTS GLOSSARY MAIN MENU FORMATS SOFTWARE INTERNET THE BOOK